An Effective Feedback Control Mechanism for DiffServ Architecture

WANG Chonggang (王重钢)[1], LONG Keping (隆克平)[1,2], YANG Jian (杨 健)[1] and CHENG Shiduan[1] (程时端)

[1] *National Laboratory of Switching Technology and Telecommunication Networks Beijing University of Posts and Telecommunications (BUPT), Beijing 100876, P.R.China*

[2] *Centre for Ultra Broadband Information Networks (CUBIN), EEE Department The University of Melbourne, Victoria 3010, Australia*

E-mail: {pgzhang,lkp,chsd}@bupt.edu.cn

Received March 30, 2001; revised August 27, 2001.

Abstract As a scalable QoS (Quality of Service) architecture, DiffServ (Differentiated Service) mainly consists of two components: traffic conditioning at the edge of the DiffServ domain and simple packet forwarding inside the DiffServ domain. DiffServ has many advantages such as flexibility, scalability and simplicity. But when providing AF (Assured Forwarding) services, DiffServ has some problems such as unfairness among aggregated flows or among micro-flows belonging to an aggregated flow. In this paper, a feedback mechanism for AF aggregated flows is proposed to solve this problem. Simulation results show that this mechanism does improve the performance of DiffServ. First, it can improve the fairness among aggregated flows and make DiffServ more friendly toward TCP (Transmission Control Protocol) flows. Second, it can decrease the buffer requirements at the congested router and thus obtain lower delay and packet loss rate. Third, it also keeps almost the same link utility as in normal DiffServ. Finally, it is simple and easy to be implemented.

Keywords feedback control, DiffServ, fairness, Quality of Service

1 Introduction

As a new QoS architecture, DiffServ[1,2] is flexible, scalable, and simple. A DiffServ domain comprises edge routers and core routers. Edge router performs traffic conditioning, and core router only forwards packets simply according to PHB (Per Hop Behavior). The traffic conditioner performs such functions as packet classifying, metering, marking and dropping. An incoming packet will be firstly classified according to the possible five-tuple (Source Address, Destination Address, Source Port, Destination Port, Protocol Type), then metered and marked according to SLS (Service Level Specifications) between users and the DiffServ domain, and dropped according to some buffer management mechanism at last. DiffServ has defined two service classes: Assured Forwarding (AF)[3] and Expedited Forwarding (EF)[4]. The packet-marking algorithms include srTCM[5] (single rate Three Color Marker), trTCM[6] (two-rate Three Color Marker), TSWTCM[7] (Time-Sliding Window Three Color Marker), etc. The queue management mechanism is mainly WRED (Weighted Random Early Detection) and some of its variations. Packet forwarding in a core router mainly includes simple queue management and queue scheduling according to packet colors (marked at an edge router). Through these mechanisms, DiffServ can provide some differentiated QoS to different traffic flows.

Simulations and some analysis in [8, 9] show that DiffServ can provide some differentiated QoS and it is easy for DiffServ to be deployed. But when providing the AF service, DiffServ still has some shortcomings such as unfairness among micro-flows, unfairness between the responsive flow and the

This work is supported by the National '863' High-Tech Programme of China under Grant No.2001AA121052, the Research Fund for the Doctoral Programme of Higher Education (RFDP) of China under Grant No.20010013003, the National Natural Science Foundation of China under Grant No.69972008, and the Nokia China R&D Center.

unresponsive flow, and unfairness among macro-flows with different RTTs (Round Trip Time), target rate or number of micro-flows. The reason is that there is nearly no state of micro-flows or macro-flows in core routers, and edge routers cannot know the state of DiffServ domain interior.

In this paper, we propose a feedback mechanism to solve the above problems in order to avoid aforementioned DiffServ unfairness. The paper is structured as follows. Section 2 will describe the proposed TFFC mechanism in details. Simulation and some analysis results are presented in Section 3. Then Section 4 gives some discussions about TFFC. We also consider the TFFC implementation in Section 5 and conclude the whole paper in Section 6.

2 Details of TFFC

2.1 Related Works

In recent DiffServ implementations, all packets are marked to colored packet at the ingress router and are not dropped except that there is congestion at the ingress router. If there are UDP (User Datagram Protocol) and TCP flows, DiffServ will let both of them enter the domain. TCP will cut their source sending rate when congestion happens but UDP will not. Thus UDP may obtain more bandwidth than TCP. The correct and intelligent action is to decrease the input (to the DiffServ domain) rate of UDP when congestion happens using the same way as handling TCP. Based on this idea, we design a TCP-friendly feedback control mechanism (TFFC) to improve the DiffServ performance. The proposed TFFC shares some similar aspects with the algorithms in [9–11].

[9] has promoted to introduce a feedback mechanism into DiffServ, but there is no detailed design of the feedback mechanism. [10] has proposed Network Border Patrol (NBP). NBP tries to emulate the TCP window mechanism at the network domain border. It is a little complicated. [11] has proposed the AFC (Aggregated Flow Control) mechanism to improve the DiffServ performance. AFC uses a virtual control TCP connection to simulate an aggregated flow and implements a flow control unit based on a token bucket, where the token changes according to the normal TCP AIMD (Additive Increase Multiplicative Decrease) mechanism. NBP and AFC are similar in that they all try to simulate the TCP AIMD mechanism in the edge router. The same disadvantage of them is that they will throttle flows sharply even if only one packet is dropped in the core router. This phenomenon will not happen in TFFC because it throttles the aggregated flow in the edge router based on rate change, not window change.

2.2 TFFC Algorithm

We assume the handled object of TFFC is the AF aggregated flow in this paper, although it is easy to extend TFFC for micro-flows. TFFC borrows the same probe-feedback mechanism as in [10] when computing the packet-loss-rate and round-trip-time in the DiffServ domain. But we can piggyback the probe information in data packets and decrease the bandwidth loss resulted from probe packets, which will be described further in Section 4. The main ideas in TFFC are listed as follows. The ingress router obtains the congestion state information inside the DiffServ domain using the probe-feedback mechanism. Based on the obtained state information, TFFC can control each aggregated flow's input (to DiffServ) rate using a specific control algorithm. We hope this mechanism can improve the DiffServ fairness property and keep high network utility simultaneously.

TFFC comprises four components (please see Fig.1): computing packet loss rate of the aggregated flow f (let it be $PLR(f)$), computing round-trip time of the aggregated flow f (let it be $RTT(f)$), computing the allowable input rate of the aggregated flow f (let it be $AIR(f)$) and throttling the flow according to $AIR(f)$. We use probe-feedback mechanism to compute $PLR(f)$ and $RTT(f)$ in edge routers. Then $AIR(f)$ can be computed according to the well-known TCP throughput equation[12] (please see the following Formula 1).

$$T \leq \frac{1.5 \times \sqrt{2/2} \times MSS}{RTT \times \sqrt{PLR}} \tag{1}$$

Here T is the throughput of a single TCP connection, MSS is the maximum session segment, RTT is the round trip time, and PLR is the packet loss rate.

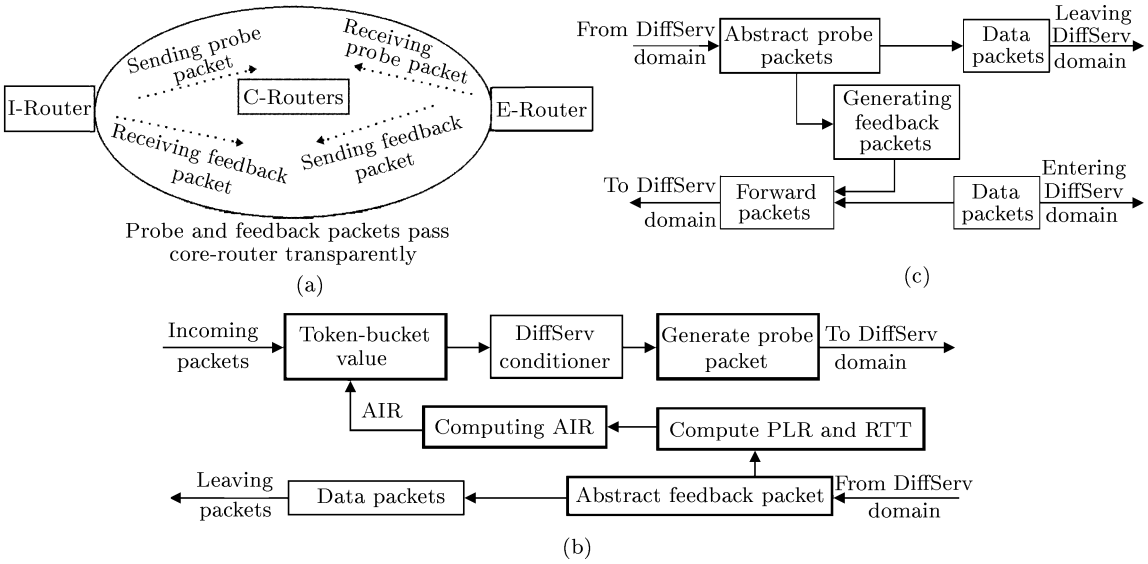

Fig.1. Details of TFFC. (a) Feedback mechanism in TFFC. (b) Ingress router functions in TFFC. (c) Egress router functions in TFFC.

After getting the $AIR(f)$ of the aggregated flow f, we can adjust its input rate at the ingress router according to $AIR(f)$. TFFC simply uses a single Token-Bucket (before the DiffServ conditioner at the ingress router) to perform this task. The token generating rate is equal to $AIR(f)$. The maximal bucket length is greater than $RTT * CIR(f)$ ($CIR(f)$ is the Committed Information Rate of aggregated flow f). When packets arrive, TFFC firstly judges whether there are enough idle tokens in the bucket. If there are enough idle tokens, the packet can enter the DiffServ conditioner and then enter the DiffServ domain. Otherwise, it will be discarded before entering the DiffServ domain.

2.2.1 Computing $AIR(f)$

In order to compute $AIR(f)$ (allowable input rate of the aggregated flow f), we need to obtain its packet loss rate ($PLR(f)$) and round trip time ($RTT(f)$) firstly. Then we can get $AIR(f)$ directly using Formula (1).

2.2.1.1 Computing $PLR(f)$

TFFC uses the similar probe-feedback mechanism (please see Fig.1(a), as that in [10]) to compute $PLR(f)$. When and only when the ingress router forwards packets into the DiffServ domain, it generates a probe packet for each data packet and forwards it into DiffServ following its corresponding data packet. Probe packets can have only the IP header whose source address is this ingress router, and the destination address is the same as that in the data packet. The core router is responsible for forwarding probe packets transparently. But the core router must guarantee that if a data packet needs to be dropped, the corresponding probe packet must be dropped, and if a data packet is not dropped, its corresponding probe packet is not dropped. When a probe packet arrives at the egress router, the egress router simply exchanges the source address and destination address in it and sends it back through the incoming interface, thus the ingress router generating this probe packet can receive its corresponding feedback packet. Moreover, in order to make computed $PLR(f)$ more precise, we can mark the feedback packet as green and try not to drop the feedback packet. In order to decrease

bandwidth loss, we can piggyback the probe packet in data packet using an IP header option: "TFFC probe option" (please see details in Section 4).

The ingress router keeps a timer in order to count the transmitted probe packet number ($TN(f,k)$) and the received feedback number ($RN(f,k)$) during the k-th time interval. The packet loss rate can be computed as follows:

$$PLR(f) = \frac{TN(f,k) - RN(f,k)}{TN(f,k)} \tag{2}$$

Because there is delay between the ingress router and the egress router, the loss rate may be non-precise and negative. We can count the transmitted probe packets using a better-granularity time scale to make the computed $PLR(f)$ more precise; the additional requirements focus on more state variables.

2.2.1.2 Computing $RTT(f)$ Passing DiffServ

TFFC uses the same probe-feedback method as that in computing $PLR(f)$ to compute $RTT(f)$. When generating a probe packet, the ingress router en-stamps the current time on it. After receiving the feedback packet, the ingress router can compute the $RTT(f)$ of this probe packet through the DiffServ domain according to the following Formula (3):

$$RTT(f,n) = \frac{RTT(f,n-1) * NumFB + CurrentDelay}{NumFB + 1} \tag{3}$$

In Formula (3), $RTT(f,n)$ is the estimated RTT at the n-th time. $NumFB$ is the received feedback packets just before now. $CurrentDelay$ is the RTT of the just received feedback packet. It needs to be said that we can compute $RTT(f)$ using the same the method as that in TCP, but we have found that its performance is worse than that using Formula (3) in our simulations.

```
At each time when AIR(f) is computed:
    NumOfToken= max (MBL, NumOfToken+ (CurrentTime− LastTime)*LastAIR(f));
    LastTime= CurrentTime;
    LastAIR(f) = CurrentAIR(f);
At each time when packet arrives
    PktLength = Length of this packet;
    NumOfToken = max(MBL, NumOfToken + (CurrentTime − LastTime) * LastAIR(f));
    LastTime= CurrentTime;
    If NumOfToken < PktLength
        Discard this packet;
    Else
        Forward this packet to DiffServ conditioner;
        NumofToken = NumOfToken − PktLength;
Note: MBL is Maximal Bucket Length
```

Fig.2. Algorithm of token-bucket valve in TFFC.

2.2.1.3 Computing $AIR(f)$

Once we get $PLR(f)$ and $RTT(f)$, we can use the following Formula (4) (derived from Formula (1)) to directly estimate $AIR(f)$ as:

$$AIR(f) = \frac{1.5 \times \sqrt{2/3} \times MSS}{RTT \times \sqrt{PLR(f)}} \tag{4}$$

2.2.2 Throttling the Flow According to $AIR(f)$

The computed $AIR(f)$ is the theoretic sending rate for the TCP window control mechanism. If an aggregated flow is responsive like TCP, its arrival rate at the ingress router will not be greater than the computed $AIR(f)$. On the contrary, if it is non-responsive or not TCP-friendly, its arrival

rate at the ingress router will be greater than $AIR(f)$. In order to make DiffServ more TCP-friendly, we can set a valve for each aggregated flow at the boundary of DiffServ (please see Fig.2). The flow rate of passing its valve is not greater than $AIR(f)$. TFFC uses a single token-bucket to perform this task. The tokens generating rate is $AIR(f)$. The maximal bucket length is MBL (larger than $RTT(f) * CIR(f)$). When an aggregated flow's packets arrive, TFFC judges if there are enough idle tokens in its corresponding bucket. If there are enough idle tokens, the packets will pass the valve and enter the DiffServ conditioner. Otherwise, the packets will be discarded before entering the DiffServ conditioner. The single token-bucket is simple-to-implement and has good performance that will be shown in Section 3.

2.3 Extending TFFC to Micro-Flows

In this paper, we assume the handled object of TFFC is the aggregated flow. In fact, we can extend TFFC for micro-flows. Under this case, ingress routers need to compute $AIR(f)$ and set a valve for each micro-flow, and to keep more state variables proportional to the number of micro-flows. The additional advantage is that TFFC can improve not only fairness among aggregated flows but also fairness among micro-flows. Moreover, we can extend the TFFC idea to the Endpoint Congestion Management architecture[13].

3 Simulation Results and Analysis

In this section, we will present some simulation results and analysis for the TFFC mechanism. We have performed many simulations under several scenarios using the commonly-used dumb-bell network topology.

3.1 Simulation Setup and Performance Metrics

We use NS2.1b7[14], an event-driven packet-level network simulator from Berkeley and LBNL, for our work. Based on the DiffServ module only provided in the NS2.1b7 version, we have developed several extensions to this simulator in order to simulate our proposed TFFC mechanism. Our simulation topologies and some parameters are shown in Fig.3. The topologies are often used in a dumb-bell way. In all our simulations, we make such assumptions as: only two aggregated flows belonging to the same AF class: Sender 1 and Sender 2, only one congested link, and uni-directional traffic from Sender 1 (or 2) to Receiver 1 (or 2). In Fig.3, "X" denotes the parameter whose value will be changed in the simulations.

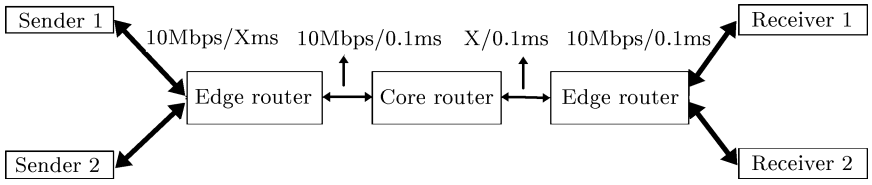

Fig.3. Simulation topology.

Our main performance metrics is fairness, called the fairness index[15]. The fairness index, FI, is defined as follows: if there are n concurrent connections in the network and the throughput achieved by connection i is equal to x_i, $1 \leq i \leq n$, then:

$$FI = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \times \sum_{i=1}^{n} x_i^2} \tag{5}$$

The fairness index always lies between 0 and 1 for non-negative throughputs, and as explained in [15], it is equal to (k/n) if k of n connections receive equal throughput and the remaining do not. Thus FI cannot be less than $1/n$ in a network with n connections.

We have also collected buffer change trajectory and congested link utility, but we omit the simulation results of congested link utility here because there is hardly any difference between TFFC and the normal DiffServ.

3.2 Simulation Results

The following are some simulation results and our analysis: the fairness of TFFC, the buffer requirement under TFFC, the additional benefits of TFFC. Under all cases, we compare the results of TFFC with those of normal DiffServ. It needs to be claimed that we omit the link utility results because we have observed that there is no apparent difference (increase or decrease) with (or without) TFFC.

3.2.1 Fairness Index

In this subsection, we mainly research the TFFC fairness between the responsive flow and the unresponsive flow. The congested link rate is changed from 0.5Mbps, 0.8Mbps, 1.0Mbps, 1.5Mbps, to 2.0Mbps. The Sender 1 including only one UDP flow (rate = 2.0Mbps) and Sender 2 with variable TCP connection are two aggregated flows (belonging to the same AF class) with the same CIR (=0.5Mbps) (Committed Information Rate). The TCP connection number in Sender 2 is changed from 1, 2, 4, 8, to 10. The results are shown in Fig.4.

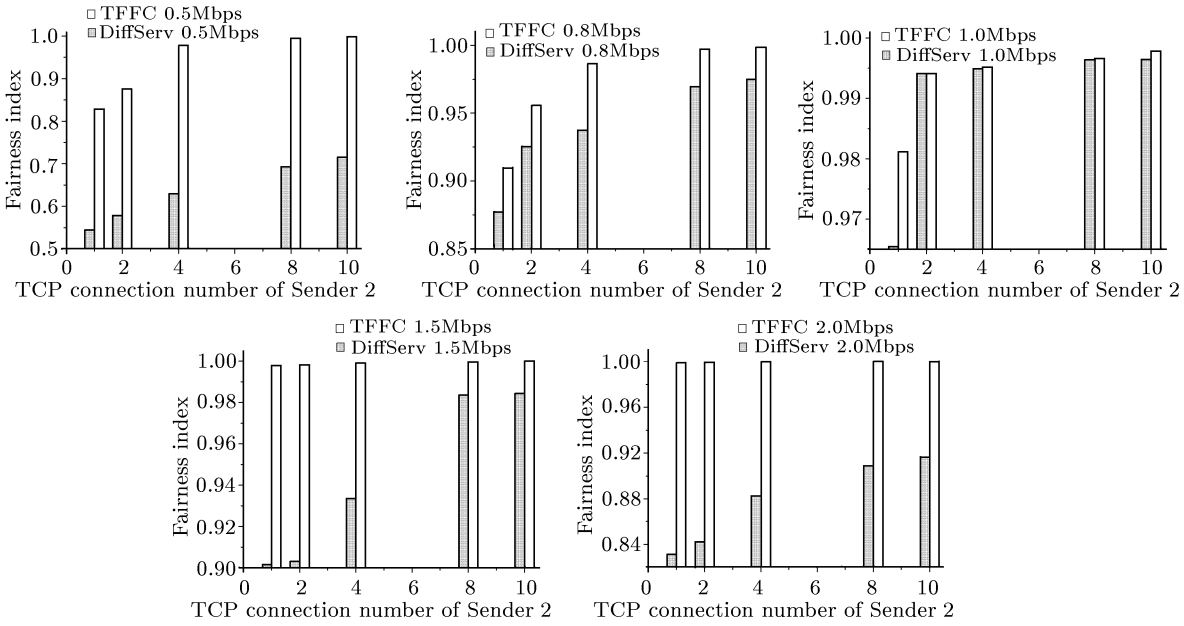

Fig.4. Fairness of TFFC and normal DiffServ.

Fig.4 shows that:

1) Under each environment, TFFC has better fairness;

2) TFFC and the normal DiffServ have almost the same fairness when the congested link rate is equal to 1Mbps, because the congested link rate is equal to the sum of Sender 1's and Sender 2's CIRs at this time;

3) Under the normal DiffServ and TFFC, when the connection number of Sender 2 increases, the fairness will increase, because more TCP connections have more strong ability to contend bandwidth with the unresponsive flow (Sender 1);

4) Under an over-provisioned network, the improved fairness under TFFC is more apparent, because there is more residual bandwidth in this case.

Moreover, we also collect the changing curve of $AIR(f)$ which will show the stability of TFFC. The simulation environment and parameters are the same as the above. Because of the limited space for the paper, we only give one of the simulation results in Fig.5 (Sender 2 has only two TCP connections, and the congested link rate is 1.5Mbps).

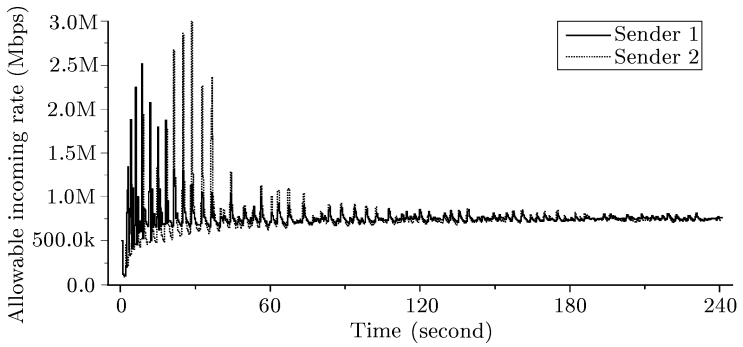

Fig.5. Trajectory of $AIR(f)$ under TFFC.

Fig.5 shows that $AIR(f)$ of Sender 1 and Sender 2 is gradually equal as time going on (because both their CIR are equal to 0.5Mbps), which proves that TFFC is stable.

3.2.2 Buffer Requirements

In this subsection, we mainly study the buffer requirements under TFFC. The simulation environment is the same as in Subsection 3.2.1. We collect the instantaneous queue length and the average queue length at the congested link for each simulation. We find out that the instantaneous queue length and the average queue length are smaller under TFFC than those under the normal DiffServ. Because of the limited space for this paper, we give only one of the results (Sender 2 has only two TCP connections, and the congested link rate is 1.5Mbps) in Fig.6. The second picture in Fig.6 is an enlarged part of the first picture in Fig.6.

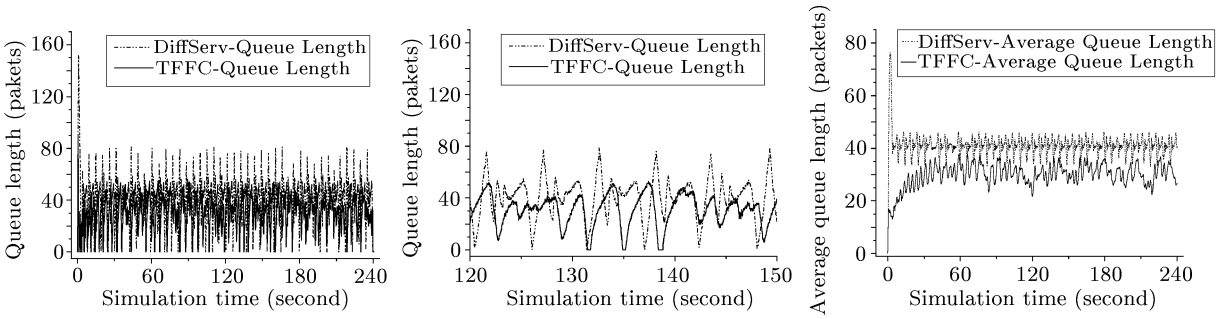

Fig.6. Buffer requirements under TFFC and the normal DiffServ.

Fig.6 shows that:

1) TFFC has less buffer requirements (decreased by about 20%);

2) Less buffer requirements will bring in smaller delay and a lower packet loss rate through the DiffServ domain.

3.2.3 Additional Benefits

Scenario 1. Impacts of Different TCP Connections

In this scenario, we mainly study the TFFC ability to reduce the unfairness between TCP aggregated flows resulted from different TCP connection numbers. The congested link rate is changed from

0.5Mbps, 0.8Mbps, 1.0Mbps, 1.5Mbps, to 2.0Mbps. The Sender 1 including two TCP connections and Sender 2 with variable TCP connections are two aggregated flows (belonging to the same AF class) with the same CIR (=0.5Mbps). The TCP connection number in Sender 2 is changed from 1, 2, 4, 8, to 10. The results are shown in Fig.7.

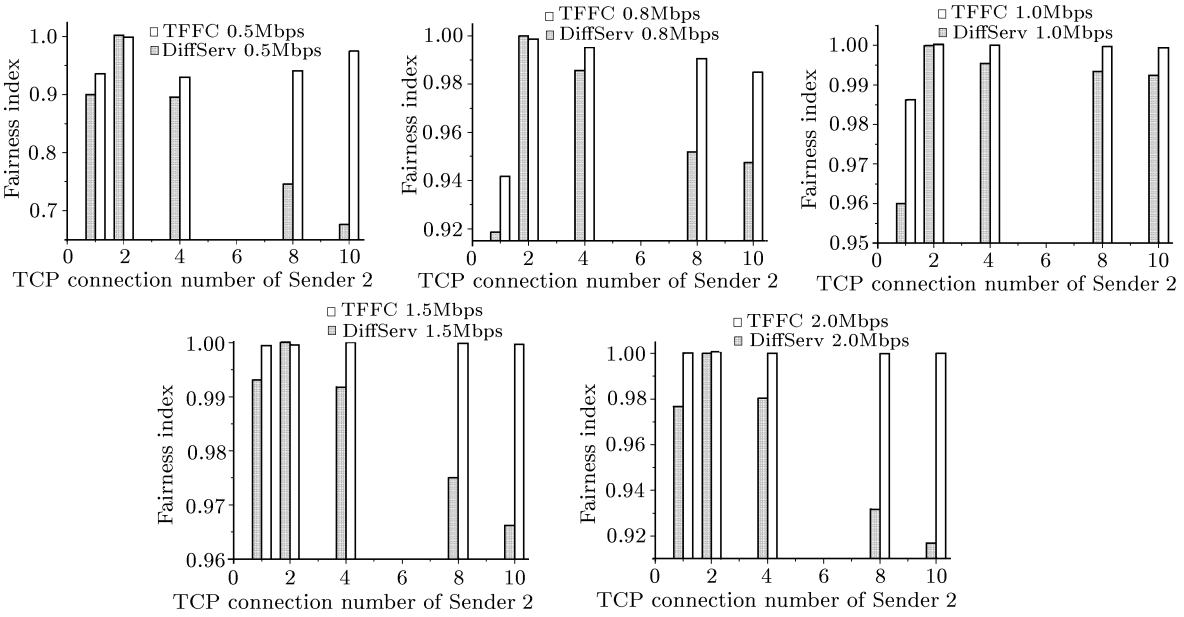

Fig.7. Fairness under different TCP connections.

Fig.7 shows that:

1) TFFC does improve the fairness between Sender 1 and Sender 2 that have different TCP connections, no matter whether the network is over-provisioned or under-provisioned;

2) When Sender 2 has only two TCP connections, TFFC and the normal DiffServ have the best and almost the same fairness, because Sender 1 and Sender 2 have the same TCP connection number at this time;

3) When the congested link rate is equal to 1Mbps (also equal to the sum of Sender 1 and Sender 2's CIRs), the fairness improvement under TFFC is not apparent, because the congested link just accommodates the total green packet of Sender 1 and Sender 2 in this case;

4) Under the normal DiffServ, the fairness decreases when the connection number of Sender 2 increases, because more micro-flow will grab more bandwidth from Sender 1 with only two TCP connections.

Scenario 2. Different RTTs

In this scenario, we mainly study the TFFC ability to reduce the unfairness among TCP aggregated flows with different RTTs. The congested link rate is 0.5Mbps, 1.0Mbps, 1.5Mbps, 2.0Mbps and 2.5Mbps. The Sender 1 and Sender 2 are two aggregated flows (belonging to the same AF class) with the same TCP connection number (=3) and CIR (=0.5Mbps). The link delay between Sender 1 and its neighboring edge-router is 1.0ms, but the link delay between Sender 2 and its neighboring edge-router is 40ms and 140ms.

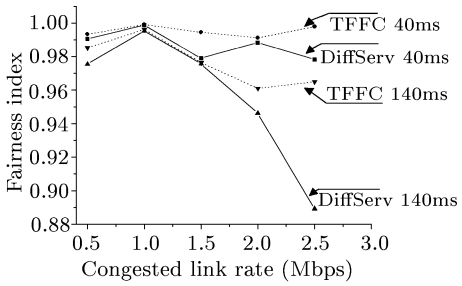

Fig.8. Fairness under different RTTs.

Fig.8 shows that: TFFC does improve the fairness between Sender 1 and Sender 2, which have different RTTs, no matter whether the network is over-provisioned or under-provisioned.

4 Some Discussions

From the above simulation results, we can see that TFFC has the following advantages: improving the fairness among aggregated flows, decreasing buffer requirements, keeping the high link utility, and making DiffServ more TCP-friendly. At the same time, TFFC still has some weaknesses such as: the impreciseness of $RTT(f)$ and the $PLR(f)$, and the bandwidth loss resulted from probe and feedback packets.

4.1 About $RTT(f)$

Although the computed $RTT(f)$ in TFFC is not the real round-trip time of a flow passing the entire network, yet we can use it approximately because the computed $RTT(f)$ is smaller than the factual round-trip time (thus $AIR(f)$ is greater than the factual allowable incoming rate and TFFC will not throttle TCP). Moreover it can decrease the unfairness among aggregated flows resulted from different RTTs to some degree.

4.2 About $AIR(f)$

Another weakness is that $AIR(f)$ is suitable for a single connection, but an aggregated flow will comprise many connections in fact. However, this factor will not influence the TFFC performance as can be seen in the simulations. Moreover it can decrease the unfairness among the aggregated flows involving different micro-flow numbers.

4.3 Bandwidth-Loss in TFFC

Finally, TFFC introduces some bandwidth loss resulted from the probe-feedback mechanism and extra burdens to network nodes. This problem can be weakened using the following methods. 1) We can piggyback the probe packet in the data packet using a special IP header option: named as "TFFC probe header" (please see Fig.9 and next Subsection for details). 2) We can compute the $PLR(f)$ and $RTT(f)$ at the egress router and feedback them to the ingress router, thus decrease the bandwidth loss resulted from the feedback packet. 3) We can generate the probe packet for every $N(>1)$ data packets, thus get a trade-off between the preciseness of $PLR(f)$ and bandwidth loss. Moreover we think that the extra bandwidth-loss and burden are worthy compared with the TFFC advantages. We will try to design an advanced method to measure the packet loss rate accurately and effectively in our future research work.

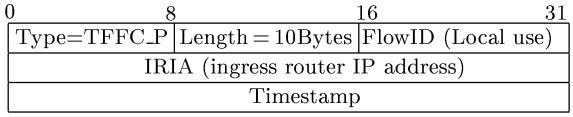

Fig.9. IP header option for TFFC probe.

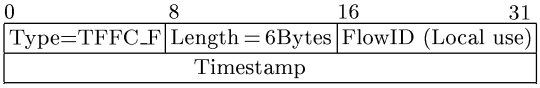

Fig.10. IP header optin for TFFC feedback.

4.4 Comparison Between TFFC and WFQ

WFQ[16] (Weighted Fair Queuing) can provide deterministic QoS guarantee (such as delay and bandwidth) for each micro-flow. WFQ is often used under the IntServ[17] (Integrated Service) architecture, and it is needed to keep the per-flow state information at each router. Although TFFC can improve the fairness properties among AF aggregated flows, yet it still guarantees only differentiated QoS, not deterministic QoS. It is obvious that IntServ with WFQ provides better QoS guarantees than DiffServ or TFFC, but DiffServ with TFFC is more scalable and simple than WFQ, and provides assured QoS acceptable to some degree. So we emphasize only performance comparison between TFFC and the normal DiffServ, and do not compare TFFC with WFQ through simulations.

```
Initialization:
    Set k = 1; /* first interval to compute PLR(f) */
    Set TFFCTimer = value; /* the time interval to compute the packet loss rate */
    NumFB = 0;
    TN(f,k) = 0;    RN(f,k) = 0;
    RTT(f,k) = 0; AIR(f) = PIR(f);
Whenever receiving a packet P from the outside of the DiffServ domain:
    Get the FlowID of packet P (assuming FlowID = f);
    Perform the TFFC value according to the algorithm in Fig.2;
    If it is needed to drop packet P
        Do nothing;
    Else
        TN(f,k)++;
        Get the current time;
        Form the TFFC probe option according to Fig.9;
        Insert the formed TFFC probe option to packet P;
        Forward packet P according to the original DiffServ;
Whenever receiving a packet P from the inside of the DiffServ domain:
    If packet P is a TFFC feedback packet
      Get the FlowID of packet P (assuming FlowId = f);
      RN(f,k)++;
      NumFB++;
      Get the Timestamp in packet P (assuming Timestamp = SendTime);
      CurrentDelay = CurrentTime-Sendtime;
      Compute the RTT(f,k) according to Formula (3);
    Else do nothing;
Whenever TFFCTimer expires:
    For each aggregated flow f
    {
      Compute PLR(f) according to Formula (2);
      Compute AIR(f) according to Formula (4);
      k++;
      RN(f,k) = 0; TN(f,k) = 0;
    }
```

Fig.11. Pseudocode for TFFC at ingress routers.

```
Whenever receiving a packet P from the inside of the DiffServ domain, from interface IF
    Get the FlowID from the TFFC probe option in packet P;
    Get the Timestamp from the TFFC probe option in packet P;
    Form the TFFC feedback option according to Fig.10;
    Get the IRIA from the TFFC probe option in packet P;
    Form the feedback packet FP;
    /* FP includes the just formed the TFFC feedback option, the destination IP address is IRIA
       in packet P, and the source IP address is the egress packet address. */
    Forward FP through interface IF;
```

Fig.12. Pseudocode for TFFC at egress routers.

5 Implementation

To implement TFFC, we need to make some changes at edge routers (including ingress routers and egress routers) and core routers. When using the TFFC probe option, we need to change only edge routers' function, and keep core routers unchanged. Figs.11 and 12 show the detailed process. We have implemented the TFFC algorithm in the NS simulator and the total additional code is only dozens of lines. In fact, we can add the TFFC function to routers through upgrading only software (not hardware or ASIC technology), because the edge router has probably a lower packet-forwarding speed. Using the Network Processor Unit (NPU)[18] technology, router's software upgrade and function extension can be realized easily.

5.1 How to Generate the Probe Packet and Feedback Packet

We can use two IP header options: "TFFC probe option" (Fig.9) and "TFFC feedback option" (Fig.10) to realize probe and feedback processes. Through inserting the TFFC probe option to each received packet, the ingress router can perform the probe function. In the TFFC probe header, we add three fields: FlowID, Ingress-router IP Address and Timestamp. FlowID is the aggregated flow's identification that the packet belongs to. If we want to support more aggregated flows, we can extend the FlowID field to 32 or 48 bits. The ingress-router IP Address is the IP address of the ingress router (incoming interface address). Timestamp is the current time in the ingress router. The feedback packet has only the TFFC feedback option, and no data payload. TFFC feedback option has two useful fields: FlowID and Timestamp.

5.2 Ingress Router

The operations in the ingress router are as follows: adding "TFFC probe option" to each received data packet before forwarding it, receiving feedback probe packets, computing flows' packet loss rates, computing round-trip time, computing flows' allowable incoming rates, throttling unresponsive flows (please see Fig.1(b) and Fig.11). As for adding "TFFC probe option", we need to form the corresponding "TFFC probe option" according to Fig.9. As for computing packet loss rates, we need to keep three state variables: the transmitted probe packet number, the received feedback packet number, and the packet loss rate. As for computing RTT, we need to keep only one variable RTT. As for computing flows' allowable incoming rates, we can directly use Formula (4). As for throttling the incoming flows, we take a single token-bucket as a valve (Fig.2), and need to keep two variables: the current bucket length and the last time of token updating. Moreover, in order to compute packet loss rates, we need a system timer (TFFCTimer). As long as the TFFCTimer expires, we start to compute packet loss rates.

5.3 Egress Router

In TFFC, the egress router is responsible for intercepting the data packet that has "TFFC probe option", generating the corresponding feedback packet, and sending the formed feedback packet to the corresponding ingress router immediately (please see Fig.1(b) and Fig.12). The operations are simpler than those in the ingress router.

6 Conclusions

In this paper, we have proposed a TCP-friendly Feedback Control mechanism (TFFC) to improve the AF aggregated flow's fairness properties under DiffServ. In TFFC, the ingress router continuously transmits probe packets into the DiffServ domain to measure network conditions and adjust each aggregated flow's allowable incoming rate dynamically. When DiffServ is congested, TFFC will drop some packets before they enter the DiffServ domain. When not congested, TFFC will allow more packets to enter the DiffServ domain. We have run lots of simulations, which show that TFFC does improve the DiffServ performance to some degree. TFFC has the following advantages. First, TFFC can better the fairness between the unresponsive and responsive flows and make DiffServ more TCP-friendly. Second, TFFC decreases buffer requirements, and obtain such benefits as lower delay and lower packet loss rates. Third, TFFC can reduce the unfairness among aggregated flows having different RTTs and micro-flow numbers. Finally, TFFC keeps high link utility at the same time and is easy to be implemented. In fact, TFFC can be integrated with other architectures such as Endpoint Congestion Management[13] and Multi-Protocol Label Switching[19] besides DiffServ. Although there are still some weaknesses such as: probe process may use some bandwidth and there is impreciseness of flows' allowable incoming rates, we believe that TFFC is beneficial to DiffServ and worthy to be implemented. We try to design a more advanced method to measure the packet loss rate accurately

and effectively, and a new TFFC-based QoS architecture will be developed in our future research work.

References

[1] Nichols K *et al.* Definition of the differentiated service field (DS Field) in the IPv4 and IPv6 headers. RFC2474. http://www/ietf.org/rfc.html

[2] Blake S *et al.* An architecture for differentiated services. RFC2475. http://www.ietf.org/rfc.html

[3] Heinanen J *et al.* Assured forwarding PHB. RFC2597. http://www.ietf.org/rfc.html

[4] Jacobson V, Nichols K, Poduri K. An expedited forwarding PHB. RFC2598. http://www.ietf.org/rfc.html

[5] Heinanen J, Telia Finland, Guerin R. A single rate three color marker. RFC2697. http://www.ietf.org/rfc.html

[6] Heinanen J, Telia Finland, Guerin R. The two rate three color marker. RFC2698. http:/www.ietf.org/rfc.html

[7] Fang W, Seddigh N, Biswajit Nandy. A time sliding window three colour marker (TSWTCM). RFC2859. http:/www.ietf.org/rfc.html

[8] Nandy B *et al.* Intelligent traffic conditioners for assured forwarding based differentiated services networks. In *IFIP Conference on High Performance Networking* (*HPN 2000*), June, 2000. http://kabru.eecs.umich.edu/qos_network/diffserv/DiffServ_papers/papers/hp

[9] Chow H, Leon-Garcia A. A feedback control extension to differentiated services. ⟨draft-chow-diffserv-fbctrl-00.txt, .ps,.pdf⟩, March, 1999.

[10] Albuquerque C, Vickers B J, Suda T. Border network protocol. http://netresearch.ics.uci.edu/NBP/paper.html, July, 1999.

[11] Nandy B, Ethridge J *et al.* Aggregate flow control: Improving assurances for differentiated services network. In *IEEE INFOCOM 2001,* April, 2001.

[12] Floyd S, Fall K. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Trans. Networking,* Aug., 1999, 7(4): 458–472.

[13] ECM, IETF Working Group. http://www.ietf.org/html.charters/ecm-charter.html.

[14] NS — Network Simulator. http://www-mash.cs.berkeley.edu/ns/.

[15] Jain R. The Art of Computer Systems Performance Analysis. John Wiley and Sons, 1991.

[16] Bennett J R, Zhang H. WF^2Q: Worst-case fair weighted fair queueing. In *IEEE INFOCOM'96*, Mar., 1996, Vol.1, pp.120–128.

[17] Braden R, Clark D, Shenker S. Integrated services in the Internet architecture: An Overview. RFC1633. http:/www.ietf.org/rfc.html

[18] Wolf T, Turner J S. Design issues for high-performance active routers. *IEEE JSAC,* Mar., 2001, 19(3): 404–409.

[19] MPLS, IETF Working Group, http://www.ietf.org/html.charters/mpls-charter.html.

WANG Chonggang received his B.S. degree in communication engineering from the Northwest Polytechnic University (NPU) in 1996, and M.S. degree in communication and information systems from the University of Electronic and Science Technology of China (UESTC) in 1999 respectively. He is currently a Ph.D. candidate of the National Laboratory of Switching Technology and Telecommunication Networks at Beijing University of Posts and Telecommunications (BUPT). His current research interests cover IP QoS, queue scheduling and management, flow control and bandwidth management, wireless networks and optical Internet.

LONG Keping received his Ph.D. degree in 1998 from the University of Electronic and Science Technology of China (UESTC). He is currently a post-doctoral fellow and associate professor in Beijing University of Posts and Telecommunications (BUPT). His research interests include SDH/ATM network survivability, ATM/IP network performance analysis, TCP congestion control enhancement, router queue management and IP QoS mechanisms (Diffserv and Intserv). He has published over 30 research papers and has finished five key projects as a key researcher and a project manager.

CHENG Shiduan is a professor and a Ph.D. supervisor. She graduated from Beijing University of Posts and Telecommunications (BUPT) in 1963. Since then she has been working at BUPT. From 1992 to 1998 she was the head of the Switching and Network Expert Group in the National '863' High-Tech Programme by the Ministry of Science and Technology of China. She has published more than 70 papers and several books in the field of telecommunications. Her research interests cover ISDN, ATM, protocol engineering, network performance, security and survivability. Currently she is working on VoIP, VoIP/IN interworking and IP QoS mechanisms.